COMSC-210 Lab 6
Big Data With Linked Structures

GOOD PROGRAMMING PRACTICES *show / hide*

ABOUT THIS ASSIGNMENT
This assignment is the second in a series that applies data structure concepts to real-life problems involving big data. We make an attempt at solving the time issue.

After you complete this lab assignment, post the required files to the COMSC server so that you receive the proper credit for this 50-point lab. Your posted files are due at midnight of the evening of the due date indicated in the course outline. Use the "Submit your FA2015 work" link on the class website to post your file for this lab, using the **lab6** folder provided.

LAB 6a: Write A Linked Array Class Template `[ LinkedArray.h and LinkedArrayDriver.cpp ]`
Purpose. The purpose of this lab is for you to write a templated class using linked structures. It's exactly the same as the **StaticArray** template from lab 3a and the **DynamicArray** from lab 4a, except that the data array is not an array -- it's a linked structure of nodes (as explained in the lecture notes).

Requirements. Write **`LinkedArray.h`** per our lecture topic 3, 4 and 6 notes, and **`LinkedArrayDriver.cpp`** to test it. Name the template `LinkedArray`, with the data type as a template variable (as in the lecture topic 4 and 6 notes). Write the public interface exactly the same as StaticArray and DynamicArray, so that they are interchangeable in any application.

You decide upon the default initial capacity to use in the template.

Write a test driver named **LinkedArrayDriver.cpp**, testing all functions, ifndef, object copy, and object assignment. Test its expandability by using an index greater than the capacity, and see what it does to size and capacity. Test its constructor's default parameter by creating and testing two LinkedArray objects -- one with no parameter, defaulting to the default initial caacity, and one with an initial capacity set in the test driver.

Submit the two files (1 CPP and 1 H) to the class website for credit.

LAB 6b: Big Data Solutions `[ DvcSchedule6b.cpp ]`
Purpose. Labs 5a and 5b took a long time to run -- it's one of the problems dealing with big data. To solve the problem, we need to use methods other than the methods we normally use to write aplications like those in lab 5.

Requirements. Write **`DvcSchedule6b.cpp`** to read and parse the 70,000 line dvc-schedule.txt text file, and find each subject code in the file. It should work ***exactly like your lab 5a***. Use your own `StaticArray`, `DynamicArray`, and `LinkedArray` templates, in any combination you wish -- but just be sure to use `LinkedArray` somewhere in the application. Do NOT use any STL containers.

Use techniques from lecture topic 6 to make the program run substantially faster than lab 5's programs. That is, it should run in under 15 seconds on DVC lab copmuters.

Submit the CPP file to the class website for credit. If you use your own `StaticArray` or `DynamicArray` templates, there is no need to resubmit them -- your previously submitted files will be used. But if you make changes in them, resubmit their H files to this lab's folder. Do **NOT** submit the very large input TXT file.

Program I/O. Exactly the same as lab 5a.

How to pause a console program: *show / hide*

GRADING RUBRIC *show/hide*
